Virtual World Accessibility: a Multitool Approach

by

Rynhardt Pieter Kruger

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Computer Science in the Faculty of Science at Stellenbosch University

Department of Mathematical Sciences,
Computer Science Division,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Prof. L. van Zijl

December 2014

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 2014

Abstract

Virtual World Accessibility: a Multitool Approach

R.P. Kruger

Department of Mathematical Sciences,
Computer Science Division,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc Computer Science

September 2014

Computer-based virtual worlds are increasingly used for activities which previously required physical environments. However, in its mainstream form, such a virtual world provides output on a graphical display and is thus inaccessible to a blind user. To achieve accessibility for blind users, an alternative to graphical output must be used. Audio and text are two output methods that can be considered. However, when using audio, care must be taken not to overload the audio channel. Channel overloading is possible with audio since it is not a selective output medium like the visual channel, that is, the user cannot choose what he/she wants to hear. Text should also be treated as audio, since a blind user consumes textual information as synthesized speech. In this research we discuss one possible solution to the problem of channel overloading, by the use of multiple exploration and navigation tools. These tools should allow the user to shape the information provided as audio output. Specifically, we discuss the development of a virtual world client called Perspective,

enabling non-visual access to virtual worlds by the use of multiple navigation and exploration tools. Perspective also serves as a framework for tool implementation and evaluation. Finally we give recommendations for improvements to current virtual world building practices and protocols, as to work toward an accessibility standard.

Rekenaargebaseerde virtuele wêrelde word toenemend gebruik vir aktiwiteite wat voorheen fisiese omgewings benodig het. Tog verskaf so 'n virtuele wêreld, in sy standaard vorm, afvoer as 'n grafiese beeld en is dus ontoeganklik vir 'n blinde gebruiker. Om toeganklikheid vir blinde gebruikers te bewerkstellig, moet 'n alternatief vir die grafiese beeld gebruik word. Klank en teks is twee alternatiewe wat beskou kan word. Tog moet klank versigtig gebruik word, aangesien die klankkanaal oorlaai kan word. Die klankkanaal kan oorlaai word aangesien dit nie 'n selektiewe kanaal soos die visuele kanaal is nie, met ander woorde, die gebruiker kan nie kies wat hy/sy wil hoor nie. Teks moet ook as klank beskou word, aangesien 'n blinde gebruiker teks in die vorm van gesintetiseerde spraak inneem. Met hierdie navorsing bespreek ons een oplossing vir die probleem van kanaaloorlading, deur die gebruik van verskeie navigasie- en verkenningsgereedskapstukke. Hierdie gereedskapstukke behoort die gebruiker in staat te stel om die inligting wat as klank oorgedra word, te bepaal. Ons bespreek spesifiek die ontwikkeling van 'n virtuele wêreld-kliënt genaamd Perspective, wat nie-visuele toegang tot virtuele wêrelde bewerkstellig deur die gebruik van meervoudige navigasie- en verkenningsgereedskapstukke. Perspective dien ook as 'n raamwerk vir die ontwikkeling en evaluering van gereedskapstukke. Laastens verskaf ons voorstelle vir verbeteringe van die boutegnieke en protokolle van huidige virtuele wêrelde, as eerste stap na 'n toeganklikheidsstandaard vir virtuele wêrelde.

This work is dedicated to my parents, who taught me what I needed to know, long before I started programming.

Acknowledgements

I would like to thank my supervisor. This work would not have been possible without your guidance and advice. And, I will never write “its” as “it’s” again.

I would also like to thank MIH for providing the funds necessary to pursue this degree.

Lastly I would like to thank my family and friends. You all supported me and contributed to keeping my mind in balance. Whether it was by giving me something besides the thesis to talk about, or ensuring that I go to dance class.

Contents

Chapter 1

Introduction

Virtual worlds are being used today in many areas in place of the physical environment. For example, it is used in education to portray real-life locations or situations; in communication it is used to create a meeting place for people who may be many kilometers apart; and it is used to simulate events which may be costly or dangerous to attempt in the real world. Finally, it is used in gaming and recreation to simulate places and environments purely for the players' entertainment.

Current virtual world implementations use the visual display as the primary method of communicating the state of the world to the user. Audio is sometimes used as well, but to a lesser extent. As the visual display is inaccessible to a blind user, most virtual worlds are currently not accessible to the blind population. In this thesis, we investigate methods to make virtual worlds more accessible to the blind population.

In real life a blind person relies on his/her remaining four senses to obtain information about the environment. This information includes:

- audio cues, including sounds from the environment, as well as acoustic properties of objects such as sound reflections from objects (echolocation);
- touch, to gather information about the size and shape of objects; and

- orientation and mobility (O and M) skills, including using objects as navigational beacons.

Most of the above information is difficult if not impossible to obtain with current implementations of virtual worlds. Some, like O and M skills and adequate audio information, result from limitations in currently available software. The sense of touch can however not currently be fully simulated because of hardware limitations.

To enable better access to virtual worlds for blind and visually impaired users, the most natural approach is to simulate the environmental cues that a blind person might expect from his/her environment. Alternative ways must also be found to convey information which cannot be simulated due to hardware limitations, such as the sense of touch. A number of solutions has been proposed in the literature to solve this problem. However, the main focus is on either text only representations of the virtual world, or hardware devices not readily available [4, 7, 10]. Although audio has been used to some extent in making video games accessible to blind people [24], it has not been used for mainstream virtual worlds specifically.

As an aid, audio can be used in two ways. One way is to simulate audio cues the user would use in the real world such as the sound of equipment. A second way is to provide secondary audio cues not available in the real world, but which nonetheless makes information available in a 3-dimensional format[1] not attainable via text-only descriptions. Examples of such audio cues may include a beep to indicate the position of a door opening, or the name of an object played as audio from the direction of its position.

When using the audio channel, care must be taken not to overload it. This overloading occurs because the audio channel is not selective, as opposed to sight which is selective. One way of overcoming this overloading problem is to provide the user with tools which can be used to explore the environment, and thereby select the information which should be communicated over the audio channel. The problem then becomes to find the right set of tools to enable the best balance between ef-

[1] 3D audio, or 3D sound, is sound played within 3D space relative to the listener's position. The direction and distance from the listener to each sound is calculated, and HRTF (head related transfer function) filters are applied to simulate the effect that the shape of the listener's head would have had on the sound in real life.

ficiency for the user when retrieving information, and the optimal use of the audio channel.

In this research we explore accessibility to virtual worlds through the use of audio cues and textual descriptions. In particular, we explore the optimal use of the audio channel through the use of exploration tools. To this end, we describe a framework which can be used to explore different combinations of exploration tools.

We design, implement and explore a framework which serves as a viewer for Second Life and compatible virtual worlds. The viewer should be designed to be completely accessible and usable by blind and visually impaired users. The viewer serves three goals: firstly, the viewer should be usable by blind and visually impaired users to access mainstream virtual worlds. Secondly, it should serve as a framework which can be used to evaluate different navigation and exploration tools, in order to optimize the use of the audio channel. Thirdly, we use this viewer to evaluate the overall accessibility of virtual worlds, with the ultimate goal of proposing improvements to both the Second Life API, as well as virtual world building practices. This last goal is thus similar to the accessibility standard maintained by the World Wide Web consortium [25, 26], which ensures that websites are designed to be accessible to all users, including blind and visually impaired users.

In Chapter 2 a survey of previous work done towards virtual world accessibility for blind users is given. Chapter 3 discusses our implementation of an accessible virtual world viewer, which also serves as a research framework for tool development and evaluation. Chapter 4 describes our findings using the accessible viewer. In Chapter 5 further work and recommendations for accessibility practices needed to ensure a better experience for blind users using virtual worlds, are discussed. Finally, Chapter 6 serves as the conclusion.

Chapter 2

Literature Survey

In this chapter a survey of previous work related to virtual world accessibility is given. Second Life, the virtual world we intend to use for the research, is described first. We then discuss previous work done on Second Life and other virtual worlds to increase accessibility for these environments. Audio games are similar to virtual worlds, as they also employ a model of the real world. We examine the methods employed by audio games, and how they can be applied to mainstream virtual worlds. Lastly, we look at accessibility APIs, a method for making 2D environments accessible that can also be applied to 3D environments like virtual worlds.

2.1 Second Life

Second Life is an online multiuser virtual environment created and hosted by Linden Labs [12]. The server maintains the state of the world, which consists of virtual objects with content (graphics and audio) and attached scripts. Users connect to Second Life by using a viewer, which is similar in function to a web browser. When interacting or moving around in Second Life, the user's display is updated continuously to show the surrounding environment, as well as the user's avatar, which is the virtual representation of his/her character.

Content in Second Life is mostly created by the users themselves, using 3D modeling tools provided by the viewer. Once created, an object is represented in the world by a set of primitive shapes like boxes and cylinders. The shape of an object on

the user's display, as well as any other content attached to it, is uploaded by the user. The user also provides the object with a name and description, but a default name (object) and an empty description is provided by the creation tools.

2.1.1 Second Life Accessibility

White et al reported on the state of accessibility in Second Life [28]. They identified various problems with the current accessibility of Second Life, including:

- Most objects in Second Life are only rendered visually – usually no textual descriptions or sounds are attached;
- The user interface of the default Linden Labs viewer does not implement an accessibility API, making it impossible for a blind user to navigate the interface with a screen reader;
- The default operation of Second Life requires the user to interact with objects using a mouse, which cannot be accomplished by a blind person in its current form; and
- Although a blind user should be able to program objects using the Linden scripting language, the object construction tools are not accessible.

White et al suggested some solutions to these problems, such as:

- Implementing an accessibility API within the user interface of the viewer. This will enable accessibility to the user interface for screen reader users;
- Implementing alternative methods of presenting the state of the virtual world to blind users, which may be in the form of textual descriptions, audio feedback, or haptic (tactile) feedback; and
- Developing a database containing user contributed descriptions for objects in the virtual environment.

In this thesis, we build on the work by White et al. In particular, we focus on finding alternative representations of the virtual world. We also examine current

accessibility APIs, and how they can be applied to virtual worlds, as suggested in [28].

2.1.2 TextSL

TextSL is a Second Life viewer developed by Folmer et al [7]. The focus of their work is to present objects in the virtual world by the use of textual descriptions. The user controls the avatar by issuing textual commands, loosely resembling natural sentences. The authors argue that a textual representation of the world is preferred over an audio representation because of the number of objects around the avatar in a usual Second Life setting. They argue that sounds may overload the user's audio channel and that sounds cannot be as easily reviewed as a textual description.

Although our solution will not focus entirely on textual descriptions to provide virtual world accessibility like TextSL, the description engine of TextSL will contribute much to our combined solution.

2.1.3 Haptic Feedback in Second Life

Depascale et al extended the Second Life viewer to provide haptic feedback to the user by using a force feedback device like a joystick [4]. Their solution can be used by a blind person to navigate the world, as well as explore his/her surroundings.

The system described in [4] consists of two functions mapped to a joystick. The navigational system lets the user move his/her avatar by pushing the joystick in the direction in which movement is desired. The joystick provides force feedback if a collision occurs; for example, when the avatar moves into a wall. In exploration mode the operation of the joystick is analogous to the white cane used by a blind person in the real world. As the user moves the joystick, the direction of interest is calculated relative to the direction in which the cane would have been pointing in the real world. If an object is present in that direction, the joystick will vibrate with an intensity inversely related to the distance to the object.

Although the system was not tested by blind users, the methods employed can be useful as part of a combined solution. Specifically, the methods can aid in

navigation around objects. However, it is not suitable for examining objects, as the system cannot present detailed information.

2.1.4 Tactile Virtual World Accessibility

Iglesias et al described a hardware device designed to make 3D objects accessible to the blind [10]. They cite a number of uses for this device, including the application of exploring virtual worlds. The device enables the user to explore 3D objects by using two fingers. Pressure is applied on a finger which encounters an edge of the object being explored. The device thus enables the user to feel the object with two fingers.

The problem with such a hardware device is that the user needs to acquire the device in question before accessibility to virtual worlds can be accomplished. This device was designed specifically for the task and will thus be costly to produce and acquire by the user, as blind users represent a small market. This is currently the case with specialized equipment like Braille embossers and refreshable Braille displays. The use of sound to convey information about the virtual world does not have this problem, as it only uses stereo speakers or headphones, which is equipment most users will already own. The device described by Iglesias et al [10] may however be useful to users who may not be able to use sound as an output medium, for example, the deaf-blind community.

2.1.5 Audemes

Ferati et al described the use of audemes as a method of conveying information to blind users [6].

An audeme is a short duration sound (5 to 7 seconds) which can be used to identify a concept. An example would be the sound of cars to represent a street. In the context of a virtual world, objects can be represented by audemes played in 3D sound relative to the user's position. The user might for example hear leaves rustling to the left, which represents a tree in that direction.

Ferati et al demonstrated the use of audemes in an information reviewing tool for blind children. However, audemes can also be applied to other kinds of information

such as objects in a virtual world [20].

2.2 Audio Games

Audio games are games that make use of audio as the primary feedback method [21]. They are usually designed for blind players. Many audio games are developed by blind developers [21]. Some research has also been done on making mainstream games accessible to the blind [2].

Audio games make extensive use of stereo sound, and is usually played with headphones. Stereo sound serves as a horizontal axis which can be used to indicate direction. Side scrollers are common, as they can be naturally adapted to use stereo sound as an output medium. An object may be represented by a sound clip; for example, the footsteps of an enemy soldier. A footstep may start off being played very softly on the right speaker alone, indicating that the enemy is to the right and far away. As the enemy moves closer, the sound will get louder, and the sound will gradually be panned from the right to the middle, and finally to the left if the enemy is moving past.

Side scrollers are easy to represent in audio as they are mostly two dimensional. Stereo sound can be used for the X-axis, and the pitch of the sound for the Y-axis. Stereo sound is, however, not suitable for representing a 3D environment, such as that used in a first person shooter or a virtual world like Second Life. Here, 3D audio is required.

3D audio can be used to play sounds in 3D space, simulating the audio that should be heard by the user. This is usually accomplished by calculating the direction of the sound and using the direction to play the sound through a surround sound system [22]. The volume of the sound is set relative to the distance of the sound from the listener. There are also implementations which can utilize normal headphones. This is accomplished by using the stereo position of the sound to indicate direction, and applying head related transfer functions (HRTF) on the sound source, which simulates the effect that the user's head would have on the sound in the real world. From this, the brain is able to calculate the approximate position of the sound [22].

Audio games can provide possible methods of rendering a 3D scene in audio. Although most audio games make use of 2-dimensional game worlds, a few games have been created that incorporate a 3-dimensional world. Examples include Shades Of Doom (an audio game resembling graphical Doom) [8] and Terraformers (a game playable in both audio and graphics) [27].

One important difference between games and general virtual worlds is that games are designed to be challenging. Thus searching for objects and locations are often designed to be difficult, as the searching action itself is as important as the finding of the object. This is not the case with virtual worlds. Audio games also have to maintain a fine balance between giving the user enough information about his/her environment, but not enough to beat the game too quickly. In contrast, the main objective of social virtual worlds like Second Life is to facilitate communication. Thus the user should be able to find information as quickly as possible, to increase productivity.

We will now look at specific examples of audio games and the techniques they employ to enable accessibility.

2.2.1 Terraformers

Terraformers was created by Westin [27]. It is a game that uses both 3D graphics and 3D audio and is playable by both sighted and blind players.

Terraformers includes several aids that can be used by the blind player to explore and navigate the environment. A virtual sonar can be activated by a key on the keyboard. The sonar plays a sound indicating the distance to the nearest object in the direction that the player is facing. A key command can then be used to identify the object. A virtual compass is also provided and consists of a sound indicating north, as well as a key command to speak the direction that the user is facing, such as north or northeast.

The audio compass and the sonar are two innovations of Terraformers which we intend to implement as tools in our implementation.

2.2.2 PowerUp

Trewin et al describe a game called PowerUp, which is accessible to people with different disabilities [24]. PowerUp enables accessibility for blind users through the use of sounds attached to each object in the 3D environment. Commands are also provided for exploration of the virtual world or to aid in navigation.

While playing PowerUp, the user can at any time hear the name of the object that his/her avatar is looking at, by the press of a key. Keys are also provided to hear the name of the object to the left or the right of the avatar. To aid in navigation, a command is provided to turn the avatar towards the nearest object to either the left or the right. When facing an object, the user can initiate a continuous walk towards the object in question. This command will also follow the object if it is a moving object, such as a car or another avatar.

Although PowerUp is a game and not a mainstream virtual world, the innovations described by Trewin et al are applicable to a virtual world like Second Life. The directional commands in particular can enable the user to explore and interact with his/her surroundings more effectively.

2.2.3 Audio Quake

The AGRIP project aimed to make the graphical game Quake, as well as its supporting software and services like level editing and score boards, accessible to blind gamers [2]. The initial project, called Audio Quake (previously Accessible Quake) consisted of a modified Quake 3 game engine using the game data files distributed by IDSoft (creaters of Quake). It also incorporated an accessible menu system instead of the graphical menu system used by visual Quake. The program was compiled for both Windows OS and Unix-like systems. It used Microsoft Speech Application Programming Interface (SAPI) on Windows to provide additional information to blind players in the form of synthesized speech, and Emacspeak speech servers on Unix-like systems.

Audio Quake augmented the sounds assigned to objects and events provided by the original Quake, to create an accessible experience for blind gamers. Most movable objects in Quake-like non-player characters had sounds attached to them in the

original Quake, such as footsteps when they are moving. Audio Quake augmented this with sounds for when the objects are silent, with a beep to indicate the position of an enemy. They also used different beeps to indicate whether the object is above, below, or equal with the player's vertical position. Audio Quake also includes sounds to notify the user about circumstances which cannot be deduced from the sounds of objects alone. One example is a scraping sound indicating that the user is partially walking into a wall, hampering movement speed.

As noted above, the Audio Quake menu system is different from the usual Quake graphical menu system, which was not accessible. Audio Quake uses a single key command system at its entry screen: the user can, for example, press `b` to begin or `d` to hear the demo. The in-game console provided by Quake has also been made accessible, enabling the user to use the console just like his/her sighted counterparts, including entering cheat codes.

Throughout the game, narrated descriptions of events are given to the user, indicating the current situation. The user may for instance hear that the character is walking through water, or that the character is near a pit. Functions like the graphical compass provided by Quake were also replaced by an audio compass, allowing the user to hear the direction that the character is facing, by pressing a key on the keyboard.

The AGRIP project included a level editing system allowing blind users to create their own levels. Rather than an interactive application, the map editor allows the user to create maps by writing a textual representation of the map, which is then compiled into a data file compatible with the Quake engine. The program uses a textual format based on XML, allowing the user to define structures as XML tags, and specifying their properties as tag attributes. The nested syntax of XML allows users to create structures inside each other, like a staircase within a room.

AGRIP modified a mainstream 3D game to be accessible to blind users. The modifications implemented with Audio Quake can also be applied to make mainstream virtual worlds accessible.

2.2.4 Shades of Doom

Shades of Doom is an audio game created by GMAGames, a company creating audio games for the visually impaired [8]. The game is based on the graphical game Doom, but with its own background story. The player navigates an abandoned military installation and needs to shut down a failed experiment. The player has to explore several maze-like levels, the shape of which is communicated to the player by special acoustical properties. Like Terraformers, a compass is provided for the player to identify the direction that he/she is facing.

Shades of Doom uses the sound of wind to communicate the layout of the immediate area to the player. For example, wind from the front means that the area is open in that direction. The echoing of footsteps is also used to communicate open passages to the user. The user might for instance hear his/her footsteps echoing to the left, meaning that a passage to the left is available. The wind sounds, as well as the use of footstep echoes as an additional navigational aid, are two innovations of Shades of Doom which can contribute to our solution.

In Shades of Doom, objects are identified to the user in three ways. Firstly, objects like walls are presented by the absence of footstep echoes and wind in that direction. A modified footstep sound is also played when the user is about to bump into a wall, indicating that the user should stop movement in that particular direction. Secondly, objects like monsters or equipment are denoted by a continuous sound played at the position of the object in 3D space. Shades of Doom makes extensive use of different equipment (air conditioners and other machines) in each room. This results in the player being able to recognize a room by the equipment inside it. Thirdly, silent objects such as weapons on the ground, or pits the player can fall into, are denoted by a special beep played at the position of the object in 3D space. When the user hears this beep, he/she can issue a command to the game which will identify the specific object, as well as its approximate distance from the player.

2.3 Graphical User Interface Accessibility

Graphical user interface accessibility is another area that must be examined in this research. Not only for the sake of following best accessibility practices when designing the client's user interface, but also for the sake of finding analogs to 2D accessibility aspects in 3D environments. This will enable an easier transition for the user to 3D environment accessibility by using familiar concepts.

A blind user will typically access a GUI by using a software program called a screen reader. A screen reader reads out any information attached to the currently-focused component using synthesized speech, or displays it on a braille display. The user navigates through the interface using keyboard commands provided by the graphical toolkit. Key commands are also provided by the screen reader to read out parts of the interface not reachable with usual keyboard navigation. This is accomplished by maintaining a virtual focus independent from the one provided by the toolkit.

2.3.1 Accessibility APIs

Accessibility APIs are implemented by toolkits to enable assistive technologies to retrieve extra information about UI components, or to enable programmatic interaction by assistive technologies. These APIs are commonly used by screen readers to retrieve information about the current state of the graphical interface. Alternative input systems may also use these APIs to interact with the UI components on behalf of the user. Examples of accessibility APIs include Microsoft Active Accessibility (MSAA) [14], the Assistive Technology Service Provider Interface on Unix systems (ATSPI) [23], and the Android Accessibility API [1].

An accessibility API exposes the UI of the current application (the application with focus) to assistive technologies. Each graphical component of the UI such as checkboxes or pushbuttons are represented by an object maintained by the accessibility API. By querying an object, the assistive technology implementation can identify which component it is, and how to present it to the user. All objects are contained in a tree structure representing the UI hierarchy of the application.

A typical application like Notepad (the text editor included in Windows OS) may

have a tree of accessible objects as follows (see also Figure 2.1):

- the main window object, containing:
  - the menu bar, containing all menus, with each menu in turn containing all menu items of that menu;
  - the toolbar, containing accessible objects representing all toolbar buttons;
  - the textbox; and
  - the status bar.

Figure 2.1: The main window object

Roles, states, and properties are three types of information provided by accessibility APIs. roles are used to define the behaviors which may be expected from a UI component. For example a button would for example perform an action when clicked. States indicate the current state of a component. Such as a checkbox which is either checked or unchecked. Properties are used to discover additional information about a component, such as the label of a button.

To take the text box of the Notepad example above, it may have the following roles, states and properties:

Text box attributes		
Roles	**Properties**	
Enter/display text	• Position on screen (x,y)	• Size of text box (w,h)
State	• Current text in text box	• Text box cursor position
☑ Editable	• Start of text selection	• Text font type
☑ Focused	• Text size	

Although accessibility APIs are mainly used to convey information about 2-dimensional graphical user interfaces like computer desktops or web pages, the information provided by these APIs are also relevant to virtual worlds. The specific information may be different, but the types of information needed to describe objects in a virtual world are the same as those used by graphical components in a user interface. For example, a role may indicate what an object will do when the user interacts with it, as the user may not always be able to deduce it from the name of the object. States can indicate the state of the virtual objects, such as whether a door is open or closed. Properties can indicate additional information like the shape of the object or its size.

One area where virtual world objects differ from graphical user interface components, is that the position and relative placement of virtual objects are often just as important to the user as their properties, states, and roles. Although accessibility APIs will often publish the position of components to assistive technologies, a screen reader may not indicate the position to the user, as the component's name, state, and role are much more important. The way the user will interact with graphical components may also be different from the conventional mouse-driven method, making screen positions irrelevant. In a virtual world, however, objects are designed to simulate real-life objects and the user might expect to translate his/her real-life methods of discovering information to the virtual world. To this end the user may need to know the positions of objects as well as their relative placement to each other.

2.3.2 Accessibility on the Web

The World Wide Web Consortium supports accessibility on the web by maintaining two best practices standards. The Web Content Accessibility Guidelines (WCAG) [26] is a set of guidelines which should be followed when designing a web page or application which will result in an accessible product. The Accessible Rich Internet Applications standard (ARIA) [25] is both a set of HTML 5 tags, and guidelines on how to use the tags, which enable accessibility for rich internet applications such as desktop style apps like GMail, or custom controls like text boxes with autocomplete menus.

2.3.2.1 Web Content Accessibility Guidelines

The WCAG describes practices that should be followed to ensure that a web page designed with standard HTML elements is accessible [26]. A screen reader or other assistive technology program examines a web page by using the accessibility API provided by the operating system or platform. The web browser exposes both its own components, such as the menu bar, as well as the components of the web page, like paragraphs or links, via the accessibility API. The web browser can however only expose the components as they are defined by the web page, and the assistive technology can only present the information provided by the accessibility API. The result is that a website which is not designed to be accessible, will be difficult for the user to use and understand. An example is the use of an image to present a link. The WCAG recommends the use of the “alt” attribute on the image element to specify alternative text describing the image. This information will then be provided by the browser to the assistive technology via the accessibility API, resulting in a blind user being read a description of the image, and the motor impaired user being able to click on the image by identifying it via a voice command. When the “alt” attribute is not used, the screen reader will only be able to notify the user that the link is denoted with an image, but not the content of the image.

The WCAG also recommends practices to make a website accessible to users who may not need a specific assistive technology, but who may benefit from specific design decisions. The guide recommends that a web site be designed to be device and platform independent. Another recommendation is that the elements on the

page be easy to distinguish, and that it be designed with sufficient colour contrast. Blinking text should also be avoided, so as not to cause a seizure in some users. Controls on a web page should also be entirely operable with a keyboard.

The WCAG is not limited to HTML alone. The guide specifies how additional media on a page should be implemented and handled, such as PDF, Flash, and audio and video clips. PDF documents should be tags to specify the reading order of the text included in the documents. Alternative forms of information should be provided for media clips, like transcripts for podcasts and captions for videos. Components designed with Flash or Silverlight should also follow the same guidelines as for web pages, including being operable with a keyboard and having text alternatives for image content.

Many of the guidelines contained within the WCAG suggest alternative representations to media which might not be accessible to people with disabilities. Alternative representations are also needed in virtual worlds. This is true in particular for objects in virtual worlds. For example, all objects should have a clear textual name attached.

2.3.2.2 Accessible Rich Internet Applications

The Accessible Rich Internet Applications (ARIA) specifications solve the problem of making custom controls accessible on the web [25]. HTML 5 is used not only to write pages, but also applications much like one would find in a desktop environment. In such applications the developer may often use custom code to implement a control with advance functionality. One example is a document editing area which supports formatting changes and layout options. ARIA is a way for developers to assign roles to controls so that the controls react in a way that assistive technology would expect. Thus ARIA enables the browser to publish the controls correctly with the platform's accessibility API, so that the controls can be identified by the assistive technology application.

The ARIA specification was designed to enable web developers to add roles to custom components, to make it possible for screenreaders to present and identify the components correctly. A similar specification is needed for virtual worlds.

Objects in virtual worlds can also be categorized into roles. Roles can enable a user to understand the object's purpose.

2.4 Summary

This chapter provided an overview of some previous work on virtual world accessibility for blind people. Second Life and compatible virtual worlds were described, and examples were given of some activities commonly performed using a virtual world. Known accessibility problems with Second Life were described, together with possible solutions to those problems. Systems that were developed for Second Life accessibility were noted. Specifically, TextSL provides a textual interface for Second Life, and haptic feedback is provided by a modified Second Life client. These last two approaches to accessibility can form part of a combined solution. Another possibility is a hardware approach to virtual world accessibility, which enables the user to explore 3D structures with two fingers. A hardware device is however not suitable for mainstream virtual world accessibility, as the user will need to acquire the device before access can be achieved. Lastly audemesare considered, as an audio based approach of conveying information about objects.

A review of audio games, and the techniques commonly employed for conveying information in audio, was given. Audio games are relevant to study in the context of virtual world accessibility, as many audio games employ a virtual world model which is similar to the one utilized by social virtual worlds. It was noted that games are designed to pose a challenge to the user, while social virtual worlds are not. Techniques used in specific audio games, including Shades of Doom, Audio Quake, and PowerUp, were described.

Finally, accessibility as implemented for graphical user interfaces and the web was described. The attributes of an accessible component, including its properties and states, were noted. The importance of the tree structure in which accessible components are contained, was highlighted. Two standards designed to connect content on the web with graphical user interface accessibility APIs were described, namely, the WCAG and the ARIA standards. The WCAG is a set of best practices which, when followed, results in useful information for accessibility tools. The ARIA stand-

ard on the other hand, serves as a method for translating custom components on the web to accessible components.

Chapter 3

Design and Implementation

3.1 Background

The objective of this research is to develop a framework which will aid in the evaluation of exploration and navigational tools. The use of tools enables the user to select relevant information to hear, which eases the load on the audio channel [21].

Our solution was developed as a framework in the form of a Second Life and OpenSim virtual world client. The client (called Perspective) will allow the investigation of different exploration and navigation tools, and the evaluation of their contribution to virtual world accessibility. Specifically, a framework will allow:

1. Exploration of different accessibility tools to find the best combination for optimizing the load on the audio channel;
2. Evaluation of the accessibility of virtual worlds and guidance towards a best practices standard to increase information available to the blind user, similar to the W3C standard for accessibility on the world wide web [25].
3. Provision of an accessible way for blind users to use mainstream virtual worlds.

An implementation of a framework of this nature should conform to a strict specification. This specification will ensure that the program is usable by its main

target audience (namely blind virtual world users and developers of that field), as well as being an effective research tool. A number of requirements to meet such a specification is outlined below.

As noted above, the framework should be designed such that exploration and navigation tools can be evaluated. This requires that the tools be implemented as modules on top of the core framework. This modular approach will allow for easy adaptation and modification of tools. It will also enable tools to be switched on or off for the evaluation of specific combinations of tools.

The framework should also enable the end-user to enable or disable tools without any knowledge about the implementation of the client itself. Hence, a configuration utility should be provided as part of the user interface for tools to be enabled or disabled.

As tools should be developed on top of the framework, the framework should also serve as an API for tool implementations. Core functionality that may be utilized by multiple tools should be implemented as part of the core framework and be available for use by other modules.

One of the core functionalities provided by the framework should be a method of playing sounds in 3D audio. Tools could then be able to utilize this to convey information more effectively to the user. When 3D sound is used to convey information about an object, the user will be able to determine the object's position by the direction and perceived distance of the sound.

As the program is targeted at blind computer users, it should be possible to control it entirely using the keyboard. Blind users do not use the mouse for interacting with the computer, and thus keyboard operation is more natural for them [3].

The client should be able to connect to both Second Life and OpenSim [18] servers. Second Life is currently one of the most popular and widely-used virtual worlds. OpenSim, on the other hand, is more cost effective to use for research purposes, as it can be hosted locally and does not require the user to hire Second Life land. Many virtual conferences are also hosted with OpenSim.

The core of the program should be user interface independent. This will make it easier to port the program to other platforms, like a mobile platform or the browser,

at a later stage. The non-graphical nature of the proposed design will lend itself to be ported to devices with limited graphical capability.

The program should present information both textually and through the use of audio cues. Audio cues are fast to understand and use, and are thus easy to use for navigation in a virtual world. Text, on the other hand, is useful for more detail about an object such as its exact size, or the spelling of an avatar's name.

The user should be able to read the textual output of the program letter by letter. This is necessary to enable the user to verify the spelling of a particular word like a name, or an address in a chat message.

3.2 Implementation

The Perspective implementation serves as both an accessible virtual world viewer, as well as a framework for developing different exploration and navigation tools. Information about the virtual world is conveyed using synthesized speech and audio cues. The user provides input to the program as either keystrokes or commands prefixed with a slash (/). Keystrokes may either be single key presses, or a combination with one or more modifier keys. Two types of key commands are available at all times:

- Reading keys enable the user to review previously spoken information by discrete units. Units may either be lines, words, or individual characters. These key commands stay constant throughout the execution of the program; and
- Action keys enable the user to control the avatar in the virtual world, or to modify the operation of the program. These keys may vary depending on the current action of the user. Arrow keys may for instance be used for moving the avatar in the virtual world, but may also cycle through menu options while a menu is displayed.

The program makes use of libraries for input and output. The program utilizes the libOpenMetaverse library [17] to connect to Second Life and OpenSim based virtual worlds. The libOpenMetaverse library is also the library used by the OpenSim

project itself. Speech is generated by using Microsoft SAPI [15] on Windows, and the Espeak software synthesizer [5] on other platforms. Graphical output, keyboard input, and 3D audio playback uses the Simple and Fast Multimedia Library (SFML) [9].

3.2.1 Components of the Program

The program consists of the components described below. For an overview, see Figure 3.1:

- User interface – maintains communication with the user. At present, both a text mode and graphical mode user interface is implemented;
- Terminal – exposes input and output routines in a user interface independent way. Used by program to receive input from the user, or display text on screen (and convert it to speech) irrespective of the current user interface;
- Client – maintains the connection to the virtual world. Instantiated by user interface;
- Several components encapsulating specific functionality of the client including:
  - Object – describes objects received from virtual world;
  - Terrain – exposes information about the terrain;
  - Sound – manages the playing of in-world sounds;
  - Chat – manages the receiving and sending of chat messages;
  - Mobility – enables avatar movement;
  - Voice – encapsulates voice communication; and
  - Creation – functionality for creating and modifying objects in world.
- Several “Perspective Modules”, each implementing a specific exploration tool. Modules can be enabled and disabled at run time;

- Several commands – contain code executed when the user types a specific textual commands. Commands can also be bound to a single-key keystroke. Commands can be created by Perspective modules, enabling tools to be controlled via commands.

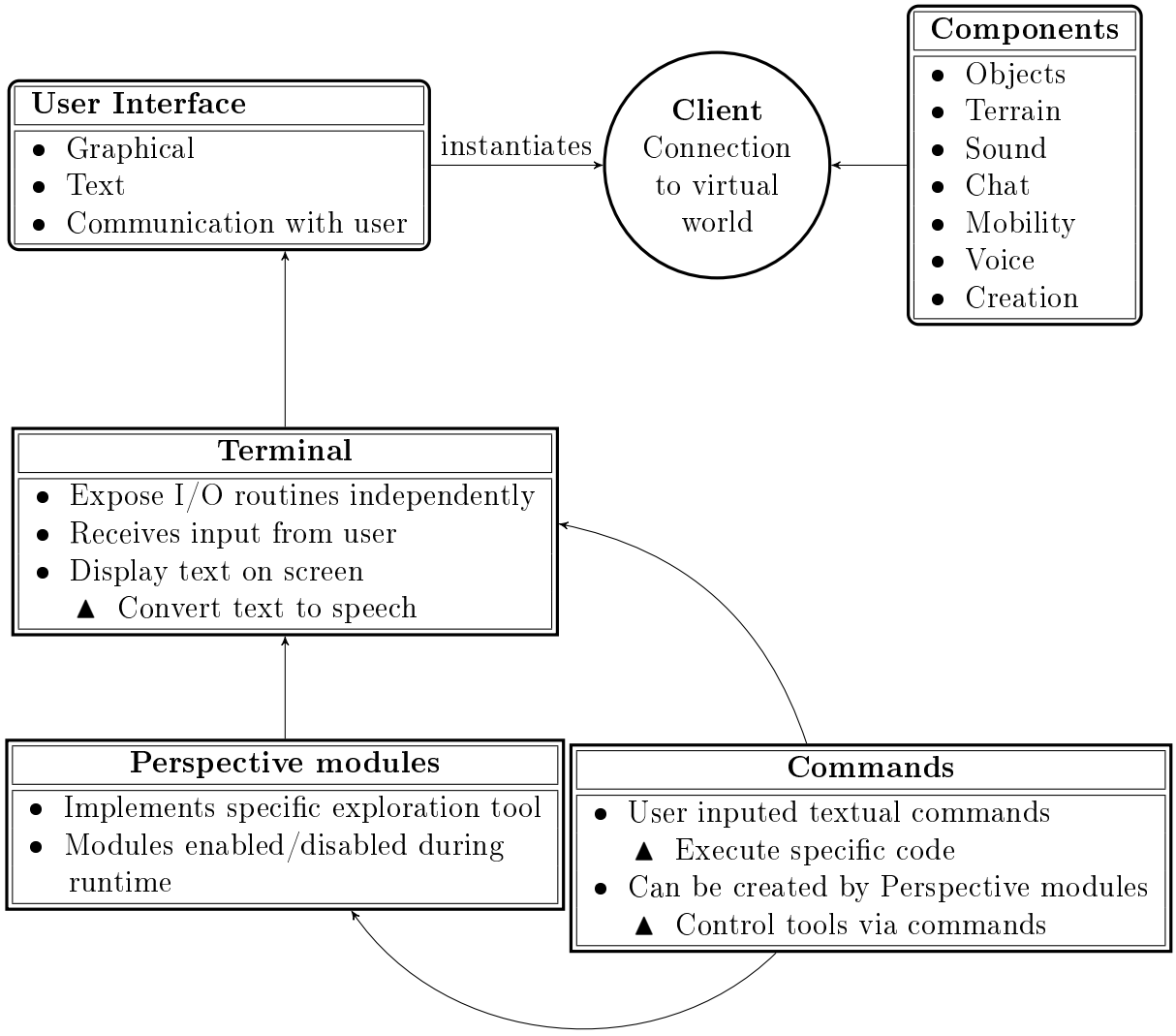

Figure 3.1: Components of the Perspective viewer

3.2.2 Program Execution

After invocation, the behaviour of the program depends on whether arguments are passed to it on the command line. If this is the case, the program directly logs in to the virtual world specified by the arguments. If not, the program displays a user interface enabling the user to enter the virtual world address and login information. With the graphical user interface, synthesized speech is used at this point to speak all information, and reading keys are available to the user. After the login procedure, the client is instantiated. It opens the connection to the virtual world and initializes all components. The saved settings are read from the configuration file, and Perspective modules are loaded according to the configuration. At this point the user is in control of the avatar, and all built-in commands as well as dynamic commands are available to the user. This includes the command to read the help documentation (bound to `F1`) and the command to end the execution of the program (bound to `q`). Several commands will open a menu to receive further instruction from the user. When a menu is displayed, the commands available to the user differ according to the type of menu. All menus allow the user to cycle through options by using the up and down arrows on the keyboard. Editable menus enable the user to change the value of each option by pressing the enter key on an option and entering a new value. FOr example, these menus are used in the settings editor and the object editor.

3.2.2.1 Focus

When using a graphical viewer, the user typically interacts with objects by using the mouse. An object is manipulated by clicking on it with the mouse, additionally clicking a toolbar button or selecting an option from a context menu. Since the Perspective viewer is a non-graphical client entirely driven by the keyboard, there must be an alternative method for interacting with objects. One way would be to let the user select the object from a menu of nearby objects each time an action is performed. The user would for instance press a key indicating that the avatar should sit, and the program prompts the user for the desired object to sit on. The problem with this approach is that the user may need to do several actions on the same object, such as examining an object, moving to it, sitting on it, and clicking it to adjust it. Folmer et al [7] handle this problem in their TextSL viewer by the

structure of textual commands. As TextSL is a text based client, all actions are performed with textual commands. The name of the object is always an argument of a command; for example, "describe chair" to describe an object named "chair". However, this would not be desirable for the Perspective viewer, as Perspective is driven by keystrokes as well.

Perspective handles this problem by maintaining a reference to an object, called the focused object. The focus acts like a cursor or highlighted area in a dialog box. When any action is performed, such as sitting or clicking, it is performed on the object in focus. Several commands are available to the user to change the focused object. The user may select the focused object from a menu, or search for an object containing a certain piece of text in its name. Commands are available to turn right or left to the nearest object in that direction, and to automatically make it the new focused object. The focus can also be continuously set to the nearest object to the user's avatar, by using the Auto Focus module (see Section 3.3.4). Lastly, the user can also change the focused object when exploring the world with the grid explorer (see Section 3.3.13).

When the user had selected the desired object as the focused object, several actions can be performed on that object. A key command will move the user's avatar closer to the object. The user can then click on the object with the enter key, which will perform an action specified by the implementation of the specific object. This may for instance open a door with most door implementations. The user can also decide to sit on the object if it is a chair; this will also activate a teleporter. A key command will give the user detailed information about the object, including its description, exact position, size, price and so on. The user may also buy the object if it is for sale. If the object is an avatar, the user may direct that a private message should be sent to that avatar. Finally the user can move the object to his/her inventory if it is an object that may be taken.

3.2.2.2 Object Labels

Within the Second Life protocol, each object is uniquely identified by its Universally Unique Identifier (UUID). In the Perspective viewer, however, the name of the object is used to identify the object to the user. A problem with this name-based

approach is that not all objects are named; some may have a generic name like "object" or "primitive", and some objects may have excessively long names like "black hardwood chair with white cushion v1.423 by jan".

To counter the problem of object names, the Perspective viewer maintains two lists of objects that can be requested by tools in local scope, and presented to the user. The one list contains all objects, while the other contains only labeled objects. A label is defined as the name of an object, except if it is a generic name like "object" or "primitive". The list of labeled objects is filtered by removing all objects with identical names (not considering case) except the one nearest the user's avatar. The list is also filtered by removing all objects not in direct line of sight to the user. Removing duplicate objects avoids the situation where a room was formed by many wall objects, each called "wall". Hence the user will only be notified of the wall nearest to his/her avatar. The other walls will be presented to the user as he/she moves around and they become the new nearest wall object. Removing occluded objects will also keep the user from trying to interact with objects that are in a different room. This eases navigation, as the user may otherwise make corrections to his/her path of movement based on objects not actually relevant.

TextSL [7] presents only labeled objects to the user. Objects are also filtered so that only the nearest five objects are presented to the user. However, this approach may again lead to objects being invisible to the user, as discussed above.

3.2.2.3 Root and Component Objects

In Second Life and compatible virtual worlds, complex objects can be formed by combining two or more basic objects (called primitives) into a link set. A link set consists of a root object with one or more children. Link sets cannot be nested, so all objects are either root objects (the parent is null), or children objects (the parent is a root object). When presenting objects to the user, it is necessary to decide whether to present only root objects, or all objects. Presenting only root objects may make it easier for the user as the amount of information is reduced. Also, most important objects like other avatars are root objects. The problem with this approach is that an object may have components that are important. For example, a projector may have a button that can be clicked. An avatar which sits

on another object is also considered to be the child of that object, and would thus be invisible to the user if only root objects are considered. The TextSL viewer does not indicate the locations of all the objects in 3D space, as it is a text only client. Perspective needs to indicate the positions of all objects to give an adequate auditory image to the user. Audio games like Terraformers [27] do not have to consider this problem as the content is created to be consumed in audio format, and thus the objects present in the world are the objects important to the user.

Perspective uses a combined approach. All labeled objects are presented to the user by default, with some tools like the click module and the occlusion module presenting unlabeled objects as well. The user can however filter objects when using the scan left or scan right tools to place the focus only on root objects.

3.2.2.4 Global and Local Scope

The tools available to the user of the Perspective viewer can be divided into tools that present local information, and tools that present global information. Global scope is defined as the area spanning the whole region. In Second Life and related virtual worlds, a region is usually 255 by 255 meters. Tools that present global information will thus consider all objects in that area. Local scope is the area within a 10 meter radius of the user's avatar. Local scope is defined by the Perspective viewer itself, and is therefore not dependent on the Second Life protocol.

Using local scope, the user is informed of the objects which may be of immediate interest, and have those objects further described and examined by tools. The local scope is thus where the user's area of interest would be. Some tools, such as the grid explorer (see Section 3.3.13), work only on global scope. Other tools, such as the positional speech tool (Section 3.3.12) and the material sound module (Section 3.3.11), work only with objects in local scope. The occlusion tool works with an area smaller than local scope, as it is used to navigate around objects. Several tools work in both scopes – a list of objects can be requested in menu form for the local or global scope, and objects can also be searched in either local or global scope.

Local scope is defined with a distance based approach, to include all objects within a 10 meter radius. An alternative would have been to define a certain number of

nearest objects as local scope, for example the eight closest objects from the user's avatar. The drawback with the latter approach is that the user may be interested in an object, but is not notified about it as there are too many objects near it. For example, a counter may have eight or more bar stools before it. Using the distance based approach, there can be no blind spots. When an object is within 10 meters of the user, the user is notified of its existence, and it can be focused and examined.

We decided on 10 meters as the radius of the local scope as it coincides with the chat range of Second Life. Within Second Life, a chat message from an avatar or object is broadcast to all objects within a 10 meter radius. Using this approach, the user knows exactly which objects or avatars would receive a chat message.

TextSL [7] provides commands to examine and interact with the virtual world. Most of the commands operate in local scope, but a command is provided to locate an object in global scope. In TextSL the default local scope is also 10 meters, although this can be changed by the user. This may create the effect that an object sends a text message which the user will not receive, even though the user can perceive the object. This problem is not considered in Terraformers [27] or other audio game like Shades of Doom [8], as these mostly operate in local scope alone. This results from the game-like nature of these projects – the user should not know of objects outside his/her viewpoint, as part of a game is exploring the world. This is different with virtual worlds for social interaction and communication, where being familiar with the environment may increase productivity.

3.2.2.5 Attachments

Second Life and compatible virtual worlds contain a special kind of object called an attachment. An attachment is any object that is in any way attached to an avatar. Attachments may be shoes or clothing, a backpack, or even a beard or hair. Attachments are commonly used to completely change the appearance of an avatar from the standard human shape to another, like a robot.

The attachment feature of Second Life and compatible virtual worlds is implemented using the same method as the link set described above. That is, the avatar serves as the root of the link set, with the attachments as the children. As with a link set, every child has an offset, which specifies its position in the virtual world

relative to the root or the avatar to which it is attached. Attachments have an additional property called an attachment point, which specifies the specific body part to which it attaches. Common attachment points are the ears, skull, feet and so on.

Attachments are important to a user of a virtual world as it influences how one appears to other users. The blind user should have the capability to add and remove attachments to his/her avatar, as well as examine which attachments are currently in use. The Perspective viewer presents attachments in a menu. The user can issue a command which will present a menu of all attachments currently attached to his/her avatar. Each menu item contains the name of the attachment, the attachment point, as well as the offset of the attachment from the avatar. When this menu is presented and an item selected, the user can issue a command which will remove the attachment presented by the menu item from his/her avatar.

Two other inventory menus are also provided. The user can request a menu of all objects carried by his/her avatar, as well as other items like note cards (small documents) and media clips. In Second Life compatible virtual worlds, objects can have inventory items as well. The second inventory menu is provided to the user for requesting the inventory of the object with focus. When an inventory menu is displayed, commands can be used to "rez" an item (place it in the current region), or to delete it. A command can also be used to attach it to the user's body.

3.3 Exploration Tools

The Perspective viewer allows the user to explore the region and his/her immediate surroundings by using several exploration tools. Tools are implemented as modules, which enables each tool to be toggled on or off individually. The Perspective viewer provides a high level API complementing the libOpenMetaverse API which can be used to implement tools. Some tools are also built-in tools in Perspective, such as the audio compass (see Section 3.3.1).

Tools can be categorized as having either continuous output, or output on activation. Tools with continuous output will continuously send information to the audio channel. They will thus continuously inform the user about the state of the virtual

world. These tools are useful for notifying the user about changes that do not occur to his/her surroundings because of action on the user's part, like movement of other avatars or objects. Tools with continuous output can also be toggled either on or off, which allows the user to further control the information that is sent to the audio channel at any particular time.

Tools which provide output when activated are useful to find specific information from the surroundings, or to examine an object more closely. These tools also include the grid explorer, which allows the user to examine the entire region, without having to move his/her avatar at all. The search features are also an example of tools which provide output on activation.

Some of the tools provided by the Perspective viewer had previously been described in the literature. This includes the key commands for turning left or right to face the next object [24], and the sonar tool for identifying objects around the user's avatar [27]. Other tools we have developed ourselves. These include the grid explorer for examining the layout of the region, and the positional speech module. The use of white noise (wind) to indicate direction has not been described in the literature before, but has been used in audio games; specifically, Shades of Doom [8].

3.3.1 Audio Compass

The audio compass enables the user to identify the current heading faced by the avatar. Compass headings are given both in degrees (with north as 0, east as 90 and so on), as well as by a descriptive name such as "north-northwest." An audio compass is also available in Terraformers [27].

There are two ways in which the user can utilize the compass. When the user presses the navigational keystrokes to turn 90 degrees clockwise or counterclockwise, the new heading is spoken using the format described above. The heading will also be announced when the user uses the turning commands to face an object near his/her avatar. The second method of utilizing the audio compass is by pressing a key on the keyboard assigned to the audio compass function. This enables the user to confirm the direction that the avatar is facing at any time.

3.3.2 Turning Keys

The Perspective viewer allows the user to turn slightly (five degrees) or execute a corner-like turn (90 degrees) either right or left, by pressing the specific key for the desired direction and turn distance. This is similar to the functions available in audio games like Audio Quake [2] and Shades of Doom [8]. Commands are also available to the user for turning either clockwise, or counter clockwise, so that the avatar is facing the next object in that direction. A similar tool was described by Trewin et al [24] in their PowerUp accessible game.

For this tool, only objects in direct line of sight from the avatar are considered. The user can continuously press the keys activating the tool, which will cycle through all objects around the avatar's position, enabling the user to identify all objects in direct line of sight. Each time the user utilizes this tool, the new object faced by the avatar is identified, and its direction and distance from the avatar is spoken. The object is also focused, allowing the user to interact with the object using any of the other available commands. The user can for instance get more information about the object, or directly teleport to its position with the press of a key.

This tool also allows the user to filter the type of objects under consideration. The user can cycle through a list of available categories, and only the objects conforming to the selected category will be considered when cycling through the objects. The following categories are available:

- Any – all objects;
- Labeled – only objects with a valid name as described in Section 3.2.2.2;
- Chair – an object the avatar can sit on, such as chairs, seats, or teleporters;
- Large – only root objects; and
- Avatars – only other avatars.

3.3.3 Sonar

A simulated sonar tool is available for the user to identify the objects which are 90 degrees to the left or right of his/her avatar. Such a sonar was also described by

Westen et al [27]. When this feature is activated, all objects within line of sight of the user's avatar, and within 90 degrees of either left or right (thus similar to the field of vision of a sighted person in real life) are placed in a collection. For each object in the collection, the program will then omit a short beep at the location of the object using 3D audio, and the name of the object is spoken. Once the object's name has been spoken, the next object in the collection is considered. This tool considers only labeled objects as described in Section 3.2.2.2.

3.3.4 Auto Focus

The auto focus tool is a tool which operates continuously, and the user can toggle it on or off. This tool computes the nearest object to the user's avatar, each time the avatar moves. The focus is then set to this nearest object, and the name of the object is spoken along with the word "focused". This tool enables the user to explore the environment by walking around. Each object will be announced as the user is closest to it. If two or more objects are an equal distance from the user's avatar, the object first considered by the algorithm will be focused.

One of the drawbacks of this tool is that some objects may never be focused, as they are occluded by other objects, keeping the user's avatar from getting close enough for them to be the closest object. This may be a problem for users relying on the tool as their primary means of gathering information. The automatic focus changing may also interfere with other tools setting the focus, resulting in the user being unable to select a desired object with which to interact. An example occurs when the user is using a tool like the directional commands to change the focus. When the user then attempts to position him/herself relative to the selected object, the auto focus tool may move the focus away from the selected object. This tool may however be quite useful for new users, as it is similar to moving around in real life, identifying the closest object by touch.

3.3.5 Footstep Module

The footstep module continuously plays a footstep sound while the user's avatar is moving. As the Second Life protocol does not have the capability of notifying clients about movement, this is accomplished by continuously comparing the user's

location to the location previously stored. The advantage of this approach is that the footstep sound is played even when the user is using auto pilot, or when the user is moved because of scripts inside the virtual world.

The footstep module enables the user to verify that the avatar has indeed moved after a movement command was issued. This also allows the user to notice the sides of objects encountered during a move operation, as the footstep sound will not be played when the movement is blocked by an object. This helps the user to follow the sides of objects, in order to find an exit from a room. The user would issue the move forward operation until a wall is encountered, and then continuously move sideways while intermittently issuing the forward move operation again until an opening in the wall is encountered.

3.3.6 Directional Look Commands

Three directional look commands, operating in local scope, are to the user's disposal. These commands will speak the name of the object to the left, the front, and the right of the user's avatar respectively. These commands can be used to identify an object blocking the user's path, or to verify that the user is facing a particular object.

The directional look commands were designed as a navigational aid, as they can be used to verify that a wall is to the user's left, front, or right. The word "nothing" is spoken when no object is detected in the particular direction. This indicates that the user may move in that direction without encountering an obstruction.

3.3.7 Search Commands

The client contains two search functions for the user to find a specific object of interest. One is provided for local scope, and one for global scope. The user is prompted for a string of text, after which the string is matched to the names (case insensitively) of all objects in either local or global scope. The search results are then presented to the user in a menu form.

The search results menu allows the user to set focus to the desired object, after which the user can interact with the object. The user may for instance examine

the object or teleport to its position.

3.3.8 Object Menus

Two object menus are available to the user of the Perspective viewer. These menus allow the user to select an object of interest from a menu. One menu contains all objects in local scope, and the other one all objects in global scope. After an object is selected from the menu, focus is set to that object, and the user can then interact further with the object. This tool is similar to functionality described by Westen [27].

A similar menu is also available for all avatars in the current region. Each menu item displays the name of the avatar along with its location within the region. This serves as a quick way for the user to verify where people of interest are, and whether interaction with them is possible or not.

3.3.9 Directional Noise

The directional noise module implements a continuous tool that can be enabled or disabled by the user at will. This tool plays a continuous sound of narrow band noise for each major compass direction where no obstruction or object is present. The noise sounds are positioned 10 meters away from the user within 3D space, in the direction of the particular compass bearing. Each compass direction is denoted by a band of noise with a frequency different from the other three, and thus this tool can also be used as an alternative audio compass by recognizing the band of noise at each direction. Although this tool has not been described in the literature, a similar idea has been utilized in the game Shades of Doom [8], where wind sounds were used to indicate the direction of hallways.

3.3.10 Echo Module

The echo module implements a continuous tool which emulates echoes bouncing from objects when the user's avatar moves. This is accomplished by playing a short click at the position of each object in local scope. The tool is triggered each time the user performs one of the movement commands. The clicks are played in sequence

one after the other, with a short time delay between each one. This ensures that the audio waves are not added together, and hence wrongly perceived by the user as one single click [13, 22].

3.3.11 Material Sound Module

Just like the echo module, the material sound module plays a sound for each object in local scope after the player has moved. The sound is however based on the specific material of the object, such as wood or metal. In Second Life and compatible virtual worlds, every object has a material which effects its collision properties. The protocol makes provision for several kinds of materials, including wood (the default), metal, flesh, and plastic. The material sound module plays a sound which consists of the default collision sound of the material (as provided by OpenSim), with its attack part of its envelope removed [13, 16]. This is to ensure that the user does not confuse it with an actual collision. The sound is reminiscent of a footstep echo bouncing from the object in question.

The material sound module can also be set to play the sounds for each object in local scope continuously. In this mode the sounds will not be triggered after a move operation, but will play continuously, resulting in a soundscape that changes according to the objects in local scope while the user's avatar moves.

3.3.12 Positional Speech Module

The tool implemented by the positional speech module has not been described in the literature before. It makes use of synthesized speech samples positioned using 3D audio. The tool creates a speech sample containing the name of each object in local scope. Each sample is positioned in 3D audio at the position of its corresponding object. The module plays the speech samples in sequence, with the time duration for each sample indirectly related to the distance between its object and the user's avatar. Hence, objects near the user's avatar can be heard clearly, while objects far off are perceived as a cluster of voices.

The tool provides positional speech output for all objects in local scope up to a limit that can be set by the user. The default number of objects is eight, but it can be set to any value greater than or equal to one. The limit was implemented

to ensure that the sequence does not grow too long. Longer sequences increase the time the user has to wait to hear a specific object. Certain areas like virtual shops can have a large number of objects clustered inside a small radius, which could overload the audio channel. With this limit in place, only the nearest objects are identified while the user can hear the rest by moving around.

3.3.13 Grid Explorer

The grid explorer is a tool that aids in the user's understanding of the layout of a region. A region is divided into equally sized blocks which the user can explore with the arrow keys. This is similar to a spreadsheet, but in three dimensions. When the user first opens the grid explorer, the region is divided into eight blocks, two for each of the X, Y and Z axes. As the user moves around this structure using the arrow keys, a summary of the objects in each block is announced. The summary consists of the object names when there are three or less objects in a block, and the name of the first object followed by the number of objects when there are more than three. When the block is empty, the message "0 objects" is spoken. The user can increase or decrease the number of blocks in the grid, effectively changing the resolution of the exploration. The user can also request all the objects in a block in list form, or teleport to the center of a block.

The grid explorer makes it possible for the user to explore an area before moving the avatar to the area and trying to navigate through it. If a user knows an area beforehand, it is easier to navigate whilst visiting it, as the user can predict where the objects and obstructions will be. The user can for instance use the grid explorer to examine the floor plan of a building before entering it with his/her avatar.

3.4 Summary

In this chapter, we described our design and implementation of the Perspective viewer, which also serves as a framework for navigation and exploration tool research. In our opinion, tools are necessary as a way of minimizing the load on the audio channel, and we identified the need for a framework aiding in tool implementation and evaluation. A number of requirements to which such a framework

should conform, was given.

The specific design for the viewer was shown and motivated. The components of the program, and the sequence it follows on execution, was given. Important aspects of the program were described in finer detail, including the handling of interaction via a focused object, the handling of object identity, and the handling of object link sets and attachments. The difference between global and local scope, and the reason for the two scopes, was highlighted. The radius for local scope (10 meters) coincides with the range of a general chat in Second Life.

The tools implemented with the framework were described. The directional focus commands enable the user to focus on a specific object around his/her avatar. The positional speech module gives a continuous overview of the user's surroundings by the use of speech played in 3D audio. The directional noise module is a complement to object identification, as it indicates empty space. The grid explorer serves as a virtual audio map, which provides a way for the user to examine a region without visiting each section.

Chapter 4

Analysis of the Framework

The graphical clients available for Second Life and compatible virtual worlds are entirely inaccessible to the blind user. The program conveys the state of the virtual world to the user in the form of a 3D graphical display which cannot be read by screen readers employed by the visually impaired. Besides the output from the virtual world, all graphical clients based on the official Second Life viewer make use of inaccessible graphical components that do not expose their state to the operating system's accessibility API. This means that even basic tasks like logging into the virtual world, or reading the chat window, are impossible to perform with the use of a screen reader. Basic accessibility clients exist to allow the user to chat with nearby avatars or move the user's avatar to a specific region or location. Radigast is an example of such a client [19]. These clients are however not capable of advanced tasks like exploring the structure of the virtual world, as they were designed only to allow the user to monitor his/her chat conversations.

A few clients have been written for the blind user specifically and were described in Chapter 2. TextSL was designed as a client which communicates the state of the virtual world to the blind user in the form of text only output. Besides being written for an earlier version of the Second Life API, the textual form of the client makes it difficult to use for the new user. TextSL filters objects aggressively to minimize the text output the user has to read through, but in our view the filters result in important information being lost to the user. A version of the Second Life viewer has also been modified to include haptic feedback for blind users. This

is in the form of a vibrating joystick which can be used either for navigation, or for exploration – simulating a white cane. This client has however not been tested with blind users. It is also not possible to get the details of an object once it is encountered by the user. Furthermore, this client was built as a modification of the Second Life viewer, and thus it also does not expose the state of its UI to the accessibility API.

In contrast, the Perspective viewer allows the user to explore the virtual world in both textual and auditory form. It consists of multiple exploration tools that allow the user to explore different aspects of the virtual structure. It has also been developed as a self-voicing application, thus enabling the blind user to use all components of its user interface. The Perspective viewer uses the libOpenMetaverse library to communicate with the virtual world servers, which is the same library used by OpenSim, the server on which the majority of non-Second Life virtual worlds is built. This means that it is compatible with the latest version of the Second Life API.

4.1 Performing Tasks

One way of evaluating the suitability of our design and implementation of the Perspective viewer, is to examine how common tasks can be performed with it. A number of tasks are performed by the typical virtual world user, and the blind user of the Perspective viewer should be able to perform the same tasks.

4.1.1 Navigation

Navigation is important, as it allows a user to explore the environment. Navigation may also be employed to seek a specific goal, such as a specific location or object. For navigation to be effective, the user must be aware of his/her environment while moving around. Thus there should be a form of real-time feedback notifying the user about important objects, as well as about spaces where there are no objects (such as the locations to which the avatar can move). The Perspective viewer contains a number of feedback tools to allow the user to accomplish these tasks. These tools provide continuous output which the user can use to verify the state of

his/her environment.

The positional speech module provides real time speech output, notifying the user about nearby objects. The speech samples are positioned using 3D audio, which means that the user can quickly determine the position of the objects in question. The speech is updated in real time, so that the name of a specific object will get closer as its corresponding object gets nearer to the user's avatar, and further away as the user moves away from it.

A complementary tool to the positional speech module is the directional noise module. Where the positional module allows the user to identify the objects around his/her avatar, the directional noise module allows the user to identify spaces with no objects. This allows the user to identify empty space to where the avatar can move.

Navigation is easier if the user is familiar with the area beforehand. In real life a blind person will often walk a route with a sighted guide or O/M instructor to get an idea of the layout of the area. When the blind person visits the area alone at a later stage, he/she is able to predict the next section of the route, as well as being able to determine his/her approximate location in the area based on the previously created mental map. The Perspective viewer contains the grid explorer for that purpose. The grid explorer allows the user to build a mental map of a location by exploring the area in a grid structure. The user is able to perform such tasks as examining the floor plan of a building, as well as the approximate distances between objects. The user can also use the grid explorer to place the avatar at a specific location in the region without having to navigate there manually.

4.1.2 Interacting with Objects and Avatars

One of the main objectives of a virtual world like Second Life is to facilitate communication. For this reason, interaction with objects in the environment, as well as with other avatars, is a key prerequisite. A virtual world viewer for blind users should therefore make it possible to interact with both avatars and objects in a non-visual manner. In graphical viewers, the user interacts with objects and avatars by manipulating the on-screen representations of the objects with the computer mouse. Such a method cannot easily translate into an auditory medium. For this

reason the Perspective viewer allows the blind user to set focus to each object in the region, to enable interaction with the object in question. Several methods are available to change the focused object, and keyboard equivalents are provided for the mouse actions of a graphical viewer.

Key commands are available to the user which can turn his/her avatar either right or left so that the avatar is facing the next object in that particular direction. The focus is also set to the new object faced by the avatar. In combination with the positional speech module, the user can move to a general location where the object would be within a 10 meter radius, and then use the directional commands to turn the avatar to face the specific object, as well as set focus to it.

Once a desired object is in focus, several keyboard commands are available which perform the same actions as the mouse commands in a graphical viewer. These commands include actions to sit on an object, grab it and put it in inventory, or buying an object if it is for sale. A command is also available to perform a general click operation, as many objects are programmed to respond to a click action, such as the adjusting of clothing to fit the avatar, or the opening and closing of doors.

4.1.3 Identifying Objects

Object identification is a prerequisite for navigation and interaction. The user needs to know which objects are around him/her to successfully navigate to a specific location. The user also needs to identify an object before interaction with it can take place. Object identification allows the user to infer the behaviour of a specific object and to know which actions apply to the object.

The Perspective viewer uses the name of an object as the basis for object identification. This is the most reliable method of identification, as it is determined by the object builder. Most objects in Second Life and compatible virtual worlds do have reliable object names, but some object names are not filled in, making it difficult to identify. The Perspective viewer generally filters out objects without names, as well as objects with generic names like “object”. This may result in important objects not being visible to the user, but the alternative is that the audio channel is cluttered with objects that the user cannot identify, and thus not interact with.

The user can use the directional commands to focus on unlabeled objects by setting the tool's filter to all objects.

The directional focus tool can filter the objects it considers by a number of criteria, which makes it possible for the user to find a specific type of object more quickly. By default the tool shows only labeled objects, but an option is available to show all objects, even those with no label or generic labels. A chair category is also available to cycle through all the objects on which the user can sit. As the role of an object is not specified by the Second Life protocol, the Perspective viewer uses the click action of an object to determine whether it is a chair. This may give false positives for certain objects, like teleporters, as these objects use the sit action as well.

4.1.4 Communication

Communication is a key part of a user's activities in a virtual world like Second Life, and therefore an accessible client like Perspective must provide an accessible method for blind users to communicate with their peers. The Perspective viewer enables the user to communicate using both text and voice. The textual communication methods include general chat as well as private instant messages to a specific avatar. Voice communication is made possible by the same voice program as used by graphical viewers.

The user can send a private message to a specific avatar by first setting focus to it. This enables the user to send a private message after an avatar is focused using tools like the directional focus tool. All textual communication is available in a list that the user can review, to confirm what was previously said.

4.2 Analysis of the Tools

All the tasks described above are facilitated by the tools available with the Perspective framework. The tools are implemented on top of the Perspective framework, using the APIs of the framework. The tools should also be evaluated, and the strengths and weaknesses of each discussed.

4.2.1 Directional Focus Commands

The directional focus commands provide an effective method of focusing on objects around the user. All the directional focus commands are able to filter objects by categories, including root objects, labeled objects, chairs, and all objects. The filter mechanism is limited by the Second Life protocol in that roles cannot be inferred from objects. The chair filter is inferred by looking at the click action of the object, but more roles could be useful. Ideally the tool should be able to filter objects by more categories, including containers, doors, tables, and so on.

4.2.2 Auto Focus

The auto focus tool automatically sets focus to the object nearest to the user's avatar, and announces the new object. As it is a tool which continuously announces the nearest object, this allows the user to explore his/her environment. It may however fail to announce all objects, as certain objects are never in the immediate vicinity. This is especially true of objects above the avatar, such as a light fixture on the roof. This may pose a problem for users relying on the tool for exploration. It may also interfere with other tools setting focus, such as the directional focus commands.

4.2.3 Step Sound Module

The step sound module plays a sound of a footstep each time the user's avatar moves. The sound is absent when movement does not occur, such as when the avatar's path is blocked by an obstacle.

Two types of movement are possible in Second Life and compatible virtual worlds. Manual movement can be performed on the `X`, `Y`, or `Z` axis, in either a positive of negative direction. A flag is set by the client indicating the axis and direction of movement, and movement occurs during the time the flag is set. This mode of movement is stopped as soon as an object is encountered blocking the user's path. Autopilot enables the user to move to a specific location. Once autopilot is initiated, the user's avatar will be moved as close to the specified location as possible, and collisions with objects will be avoided.

Due to the limitations of the Second Life protocol, a footstep is not always played on movement, even if there is no object blocking the avatar's path. The protocol does not have a way of moving the avatar a set amount using manual movement – this is only possible with auto pilot. Auto pilot is not always desirable for a blind user, since it will automatically move around objects, meaning that the user will not be notified of the existence of the object. For manual movement, the protocol only allows viewers to set flags indicating movement in either positive or negative `X`, `Y` or `Z` axes. When the flag is unset, the movement is stopped; thus clients should set the flags for a specified time in order to move for a certain amount, or set the flag when the key for movement is pressed, and unset it when the key is released. Furthermore, on collision the avatar may move slightly forward, or even backward (as in a rebound). The only way for the step sound module to check if movement did indeed occur, is to look at the positions of the avatar before and after movement. The step sound module uses a threshold value to filter movement blocks and bounces. This can however not always be correct, and the module may give false information. The best approach for using the module is to press the movement keys continuously.

4.2.4 Directional Noise Module

The directional noise module plays narrow band noise continuously in the direction where no object is positioned. The middle frequency for each band of noise is different to the others; thus the noise can also be used to confirm the direction that the avatar is facing.

Due to the fact that the Second Life protocol has no way of determining an object's role, the module will not render objects facilitating movement as open, such as a door. It can also happen that the module will not play a noise in a specific direction indicating that it is closed off, when in fact only a corner or small part of a large object is blocking the path and the user may in fact be able to move around it. Thus the module only plays noise for directions where no objects are to be found.

4.2.5 Positional Speech Module

The positional speech module plays the spoken names of objects around the user at the position in 3D space where the particular object is situated. This enables the user to get an overview of the objects around him/her, without having to focus on each. As the module operates continuously, the user receives input from the module without any required action. The audio is also updated as the user moves around.

The tool continuously updates its sequence of samples based on the objects in local scope, resulting in sound feedback that changes as the user's avatar moves, or as other objects like avatars move around the user. The module also implements a command that can be used by the user to temporarily silence the positional speech output, in order to focus attention on output from another tool. While the continuous output is paused, the user can request a once off playback of the sequence by a command provided for that purpose.

4.2.6 Grid Explorer

The grid explorer allows the user to explore the whole of the current region by partitioning it into a grid of blocks. The grid starts out with a size of two blocks per axis, but the user can increase the number of blocks in order to explore the region in a higher resolution.

The grid explorer is best suited for examining objects which are rectangular in shape. Objects with curving sides are more difficult to examine. This is due to the fact that navigation through the grid is only possible along either the `X`, `Y`, or `Z` axis. Another problem is that an object might partially extend into an adjacent block, and the user might think that the whole block is occupied by the object. The user thus has to set increasingly finer resolutions to find the edges of objects.

4.3 Analysis of the Framework

One of the objectives of the Perspective viewer is to serve as a framework for the evaluation of navigation and exploration tools. The viewer should thus serve as an API which can be used to develop tools. Tools should have the capability of being

toggled on or off, to evaluate their contribution to the accessible rendition of the virtual world.

The Perspective viewer employs a modular approach to tool implementation. Tools are developed as modules which can use the API provided by the Perspective framework. API routines are available for the cross-platform generation of synthesized speech (Microsoft SAPI on Windows, and Espeak on Unix-like platforms), as well as 3D audio functionality. The user interface of the viewer is abstracted, allowing the viewer to be ported to different platforms in the future, without impacting the implementation of tools. The framework also contains accessible user interface components that can be used by tools to receive input from the user. These include text entry components and menus which are independent from the specific user interface used by the viewer. Tools are also able to register commands which the user can use to control the execution of the tool. These commands are activated by textual input, but can also be bound to keyboard keystrokes.

The Perspective framework provides API calls which can be used to examine the state of the virtual world. Modules can register event handlers in order to be notified of changes to the user's context. Modules can query objects in either global or local scope. Routines are also available to perform common tasks like calculating the distance to an object, or determining whether an object has a useful label.

The modules of the Perspective viewer are built as .NET Framework assemblies, which are dynamically loaded at runtime by scanning the module directory. This has the advantage that modules can be built by a user without access to the source code. This also aids the distribution of tools, as new tools can be distributed to users without having to re-install the viewer. The modules can be enabled and disabled by the user, and the state of each module is saved in the viewer's configuration file, and thus the state of the viewer is restored with the next execution.

4.4 User Evaluation of the Viewer

The evaluation of the Perspective viewer consisted of two parts. Firstly, the viewer was used to attend a virtual conference, where a presentation on the viewer was also given [11]. Secondly, the viewer was informally evaluated by blind users and

some of their feedback has been incorporated into the program. Ideally the program should undergo formal user evaluation, but due to time constraints this was not feasible.

4.4.1 Use at Conference

The Perspective viewer was successfully used in attending a virtual conference [11]. Stations were set up in the virtual environment, each configured for a specific presentation. The avatars moved together from station to station, and a presentation was given by an avatar at each station. The Perspective viewer was used to find the place where the avatars were meeting at first login. This was accomplished by using the avatar list, which lists the name of each avatar along with its location. The teleport function was then used to move the avatar to a location close to the meeting point. Most of the conference was conducted using voice communication, which was possible with the Perspective viewer since it connects to the same voice communication service as the mainstream viewers. Chat messages were also used by users who could not use voice communication, which was received by the Perspective viewer and read using synthesized speech. When the avatars were moving to a new station, the positional speech module could be used to hear their locations and move in the same direction. This made it possible to infer, from the direction the avatars were moving, the new station to approach. As the user moved into a 10 meter range from the new station, its objects were announced by the positional speech module. If the avatars moved out of the view of the Perspective viewer, they could be rejoined by teleport to their coordinates.

Once the avatars reached the station where the presentation on the Perspective viewer should be given, the directional commands could be used to turn the user's avatar to face the presenter's chair. This also set focus on the chair. A keyboard command was then given, instructing the avatar to sit on the chair. During the presentation the avatar was also made to stand and the directional commands were used to focus on the objects and avatars in the location, while their names were mentioned in the presentation. The sighted avatars could verify the focused object by the direction in which the user's avatar was facing.

4.4.2 Evaluation by Users

The program was provided to five blind users with the goal of being informally evaluated. Feedback has been gathered and where possible incorporated into the program. We discuss specific areas where feedback was provided.

4.4.2.1 Grid Explorer

The users were able to successfully use the grid explorer to examine an area in a region before visiting it with their avatars. The commands for navigating among grid blocks was familiar to the users as it is similar to the commands for navigating among cells in a spreadsheet.

4.4.2.2 Footstep Sounds

The footstep sound module was developed after user feedback indicated that it would be helpful to have an audio indication that the avatar has moved, every time that a movement command is issued. This is particularly important in the case of remote virtual world servers, due to a possible time lag. In addition, the Second Life protocol only makes provision for turning movement on or off in a certain direction. Therefore the distance of movement is dependent on the time between toggling the movement flag on and off, and it might happen that the user issues a movement command and movement does not actually occur, or only a short distance is traveled. The footstep sound module addresses exactly this problem, by playing footstep sounds while the avatar is moving.

4.4.2.3 Teleportation Notification

In the initial Perspective implementation, it was possible to confirm, after a teleport command was given, whether a teleportation actually took place by checking the avatar's position, and by being notified of the change of surrounding objects. User feedback indicated that a spoken message notifying the user about the teleportation would be preferable. The client was subsequently modified to notify the user about a successful teleportation by printing and speaking the message "teleport complete". When the teleportation process failed, the message "teleport failed" is spoken instead.

4.4.2.4 Auto Focus

Three out of five users selected to use the auto focus module while moving around in the virtual world, as they were automatically notified by the module about the nearest object. This also enabled them to immediately interact with the object closest to them, and change the focus by just moving to a new object.

4.4.2.5 Positional Speech Module

All the users found the positional speech module useful for a continuous notification of objects around their avatar. Some users had a problem distinguishing the speech produced by the positional speech module from the speech produced by the other tools and the client itself. To this end, a volume control was added to the positional speech module, allowing the user to set its volume to a level not interfering with other speech output.

4.4.2.6 Directional Noise Module

As in the case of the positional speech module, the users indicated the need to configure the volume of the directional noise to their preference. To this end a volume control was also added to the directional noise module.

Volume controls may be useful not only for the directional noise and positional speech modules, but for all tools implemented on top of the Perspective framework. Volume controls add a further method for shaping the information rendered with the audio channel. Individual volume controls for every tool is an area that can be explored in future work.

4.5 Summary

In this chapter the Perspective viewer was analyzed in terms of the tasks that a user can perform while using it. It was shown that the user can use the viewer to navigate, identify objects, and interact with objects. Importantly, the user can also communicate with other avatars in the virtual world.

Each tool described in the previous chapter was analyzed individually, and their limitations described. These limitations are both a result of the current Second Life protocol, as well as building practices currently employed by virtual world builders.

The Perspective viewer was analyzed as a framework for implementing new navigation and exploration tools. The Perspective framework provides an API that can be used by tool implementations to perform common functions. These functions include retrieving information about the state of the virtual world, playing sounds, and producing synthesized speech. Modules can also be built without access to the source code of the viewer, and can be disabled and enabled at runtime.

Finally, a short evaluation of the Perspective viewer was presented. The viewer was used to attend a virtual conference on virtual environments. The viewer was also informally evaluated by five blind users. Modifications that were made to the viewer as a result, were described.

Chapter 5

Accessibility Recommendations

The Perspective viewer provides improved accessibility for blind users accessing Second Life, yet the extent to which accessibility can be achieved by the current implementation is hampered by limitations in both the Second Life protocol and current virtual world building practices. This chapter seeks to review the limitations in the Second Life protocol and in building practices, and to make recommendations on how this can be improved.

5.1 Building Practices

The Perspective viewer uses the name of an object for object identification. The name is however not always filled in by virtual world builders. Furthermore, many objects have names such as simply "object", or "prim". According to Folmer et al [7], about 30% of objects in Second Life do not have a name useful for object identification. One possible reason is that the graphical client for Second Life does not show object names by default, but rather just the 3D shape of the object. Object names are important not only for accessible clients like the Perspective viewer, but also for the sighted user who either wants to search for a specific object, or in the case where the graphical display is not available, use clients available for mobile platforms like Android. These clients are often designed for chatting or the checking of instant messages. The clients often do not have a graphical display capable of displaying the 3D virtual environment, and the user is only able to teleport to a specific object by name. In this case the name of an object is also

important to the sighted user.

Our first guideline therefore calls for descriptive names for all objects. Names should be short, but clear enough so that a user can identify each object by its name.

The Perspective viewer also describes an object by making use of the description field. But this field is seldom filled in, and often used only for advertising, for example to give the name of the object developer. Again, the object description field can be useful to both blind and sighted users in that it can be used to provide instructions on using the object. Currently owner chat – a method by which an object can chat to its owner – is often used for this, but if graphical clients are written to display object descriptions on a mouse hover or other user interaction, the object description can instead be used for this purpose.

Our second guideline therefore proposes that objects should be assigned meaningful descriptions where relevant. The description of an object should also be updated as the state of the object changes.

Another reason for the lack of adequate object names and descriptions is that the building tools of graphical clients typically fill these fields by default. The fields should rather be left blank, and the user prompted for an adequate name and description. This will encourage builders to think about useful object names and descriptions.

5.2 Improvements to the Protocol

Currently the Perspective viewer uses the unmodified Second Life protocol to gather information about the virtual environment, to present it to the user. This enables the viewer to render the world adequately for most tasks the user might want to perform. Yet, a better rendition of the virtual world can be produced in audio if improvements are made to the Second Life protocol.

5.2.1 Room Identification

In Second Life and compatible virtual worlds, rooms are formed by building objects to represent the walls, doors, floor, and roof of the room to be modeled. This makes it hard for accessibility tools like the Perspective viewer to identify, as the objects are not always in the same link set, since they may be used for more than one room. One example is a dividing wall.

We propose that a method be added to the Second Life protocol and building tools so that a room can be defined as a specified area in a region. The builder should be able to specify an area as a room, after which he/she can fill in its walls and other room parts with virtual objects. This will enable accessibility tools like the Perspective viewer to indicate the current room to the user, as well as its size and if the user is near the edge. This will also have applications for other users as well. The graphical viewer may list popular rooms the user can visit, or enable scripted objects to navigate to a specific room.

Furthermore, this method could be extended so that the parts of a room can be specified; that is, which objects are its door, roof or walls. This will allow the blind user to be notified of the direction of the nearest door, or the room adjacent to the current room. This will also make it possible for builders to copy a room with all its room parts, to be placed in another location.

5.2.2 Object Roles

In accessibility APIs, the roles of components are used to indicate the type of a component, what information the user might expect from the control, and the actions allowed on the specific component. Roles are also commonly used on the web to allow screen reader users to quickly jump to a specific component on the page, such as to jump to the next or previous link, paragraph, or button. The Second Life protocol does not currently have a dedicated mechanism to indicate the role of an object, and it is not specified by builders of objects. Sighted users deduce the role of an object by its shape, as presented on the 3D display. The Perspective viewer deduces one role, that of a chair, by examining the click action of the object in question.

Defined roles for virtual objects will make it possible for viewers like the Perspective viewer to filter the objects presented to the user by role, to reduce the number of objects presented, and thus the load on the audio channel. It will also enable the user to find an object with a specific role, for instance to find the nearest door or sign post.

Object roles can also be useful for sighted users, but especially virtual world builders. Roles enable the categorization of objects by type, enabling the user to quickly browse his/her previously built or bought objects. Currently the user is able to manually organize objects in his/her inventory by folders, but object roles can enable the client to organize objects automatically, much like a music player will organize songs by genre or artist.

Roles can be implemented with either text or predetermined values, typically integer constants. Text has the advantage that new roles can be added by builders as needed. The drawback of this approach is that the roles assigned by builders may not be consistent; compare for instance the roles of closet and cupboard. Localization is also a problem in this case, since roles will typically be in English, and may need to be presented in another language to the user. The enumerated integer constants approach facilitates consistent roles, but it may not be as easy for a user to add a previously unused role to an object.

We propose a combined approach as a solution to this problem. Enumerated integer values should be used to define roles, with the characteristics of each role described by the protocol documentation. One role that should be available is the role of "CUSTOM". In addition, each role should have a textual description, which is automatically set to the English name of the role referred to by the integer value. In the case of a "CUSTOM" role, the user should be able to enter a custom label for the specific role. This provides the consistency of integer roles, but the flexibility of textual roles.

5.2.3 Object States

Roles are useful for determining the general type of object, but additional information is usually needed to give an adequate representation of the object to the user. States would allow a protocol to convey a property of an object which may have a

predetermined value. A door, for instance, may have a state denoting whether it is open or closed. A light may be either on or off.

5.2.4 Object Properties

Properties are key value pairs that give additional information about objects. A graphical user interface accessibility API might provide a property on each object indicating its position on the screen, or its content description. Second Life already provides properties for each object, and it can be queried and updated through the Second Life protocol. The name of each object is an example of a property provided by Second Life. There are however properties which are absent from the current implementation, which would increase accessibility.

The facing direction of an object is one example of a property that should be added to the protocol and protocol implementations. This is different from the object orientation already provided by Second Life. The object orientation is used by builders to rotate an object's shape in 3D space. The facing direction on the other hand should be used to specify the direction of the object's front. The facing direction of a staircase, for instance, is the direction in which the stairs are facing. In other words, it is the direction that the user would go when descending the staircase. This will also be useful for objects that moves around the virtual environment, as they will be able to approach staircases, doors, paths and other directional objects from the correct angle.

We propose that more properties be added to the Second Life protocol, as well as Second Life itself and Open Simulator. These properties should also be available in the object creation user interface, so that the builder can input relevant values.

5.2.5 Accessibility Events

Sometimes an object may change due to user interaction or according to its running scripts. These changes are often conveyed to the user by changing the visual appearance of the object, but the textual information of the object is not updated. One example of this is a door opening. Another example is a projector object with a changing slide. A method should thus be implemented by which accessible clients can be notified of such changes.

A graphical user interface may also have components that change, and this information are usually conveyed to assistive technologies by the use of accessibility events. An accessibility event may indicate for instance that the text in a text box has changed. We propose that a similar method be employed in the Second Life protocol to indicate object changes to the user. The protocol already contains some events indicating various occurrences, such as the arrival of a chat message or the addition of an object to the region. This implementation should be extended so that object builders can specify events to be sent during script execution.

Another event that should be added to the Second Life protocol is the collision of the user's avatar with an object. There is already an event indicating that an object has forcefully collided with the user's avatar, but it is not triggered when the avatar bumps into a wall or the collision is a slight one. This event will make it possible for accessible clients like the Perspective viewer to detect when the avatar bumped into an obstacle, which will enable the user to find a path around it.

5.3 Summary

In this chapter suggestions were given for the improvement of the current state of accessibility of virtual worlds. The improvement of current building practices for better accessibility was described. The issues under discussion included two properties of every virtual object, namely, its name and its description; the role of an object's name, and how it is crucial for accessible object identification; and the description property of all objects, and how it can further aid in providing a more accessible experience to blind users.

Improvements to the Second Life protocol were suggested, that would enable accessible viewers like the Perspective viewer to provide more relevant information to users. For example, rooms should be defined by builders and communicated by the protocol, as a region in virtual space, and the parts of the room should be specified. This feature can be useful to not only blind users, but sighted users as well.

It was suggested that extra information should be added to every virtual world object. Such information can be encoded in roles, states, and properties. Roles enable the protocol to specify the type of object, which when conveyed to the

user, will signify which actions can be performed on the object and what behavior expected. States specify the current state of the object, for example whether a door is open or closed. Properties are used to store additional information about the object, like the direction it is facing.

Chapter 6

Conclusion and Future Work

In this research we explored the accessibility of virtual worlds for blind users, with emphasis on the use of navigation and exploration tools. We pointed out that the auditory sense, unlike the visual sense, is not selective. Therefore, the information provided via the audio channel must be carefully filtered in order not to overload the auditory sense. We proposed to provide the user with a set of navigation and exploration tools to examine the virtual world. This method effectively allows the user to select the information that he/she wants to receive, and thus prevent audio overloading.

It was decided to focus on Second Life and compatible virtual worlds, as the Second Life protocol is the most widely used virtual world protocol. Accessible viewers for Second Life has been developed in the past, including a textual viewer [7] and a viewer providing output as haptic feedback [4]. However, the former provides output as text which is sequential, and thus time consuming to review, while the latter cannot render object details. Audio was also used as an output medium, but only for audio games such as PowerUp [24], and not for mainstream virtual worlds.

We developed a viewer for Second Life and compatible virtual worlds, called Perspective. Perspective allows blind users to use virtual worlds independently by providing output as synthesized speech and audio cues, as opposed to traditional virtual world viewers which provide output on a graphical display. Perspective serves also as a framework for tool development and evaluation, providing an API for developing navigation and exploration tools, and a method for adding and re-

moving tools.

Our threefold aim in developing the Perspective viewer was reached: firstly, it serves as an accessible viewer for Second Life and compatible virtual worlds; secondly, it serves as a platform for developing and evaluating tools; and lastly it allows us to evaluate the accessibility possible with current implementations of virtual worlds. We illustrated the usability of the viewer as an assistive technology by looking at the tasks that a user can perform with the viewer. The viewer was also used in attending a virtual conference, and an informal user evaluation was done with blind users. We analyzed the viewer as a framework by looking at the API it provides, and how it serves to aid in tool development. The viewer allows tools to be developed without access to the viewer's source code. The API provides tools with the capability of receiving updates from the virtual world as a list of objects. Objects can also be filtered according to their category and whether their names are useful to the user. The viewer also provides audio and synthesized speech playback as part of the API.

Finally, we suggested improvements for the current state of virtual world accessibility in terms of the limitations posed by the protocol and building practices. We concluded that useful names and descriptions of objects should be enforced with world building tools. We described a method for specifying room locations that should be provided to the Second Life protocol, and how it will also benefit sighted virtual world users. We pointed out how roles, states, and properties (features of graphical accessibility APIs) can be applied in the context of virtual world accessibility.

Future Work

The current implementation of the Perspective viewer is constrained by the limitations imposed by the Second Life protocol. Once the adaptations to the protocol described in Chapter 5 is implemented, the viewer can be improved with the new API to provide more information to the user. This will include more detailed information like announcing the user's current room, as well as navigational tools like finding a path from one location to another by making use of the semantic information that will be exposed by the protocol.

The current implementation of the client uses only the audio channel as output, although basic output on a graphical display is provided for demonstration purposes. One avenue for future work may involve the addition of more hardware peripherals. Touch screens are becoming common, and will allow the user to explore the world by sliding his/her finger around the screen. Coupled with vibration devices like those currently contained in phones and tablets, it will be possible for the program to simulate tactile feedback.

The Perspective framework was implemented to be platform and user interface independent. This will allow the viewer to be ported to new platforms like mobile phones and tablets. The output provided by the Perspective viewer is especially suited for mobile platforms, as it does not make use of sophisticated graphics. The audio can be produced by the 3D capability provided by platforms like IOS and Android.

Finally, the tools described in this research can be adapted to provide audio feedback to blind users while they are moving around in real life. A camera can be used to identify objects around the user, and image recognition technology can be used to infer names for the objects. Tools like the grid explorer could be adapted to allow the user to explore an area before visiting it in person.

Bibliography

[1] Android developers. Accessibility. `http://developer.android.com/guide/topics/ui/accessibility/index.html`.

[2] Atkinson, M., Gucukoglu, S., Machin, C. H. C., and Lawrence, A. E. Making the mainstream accessible: Redefining the game. In *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames* (2006), Sandbox '06, ACM, pp. 21–28.

[3] Borodin, Y., Bigham, J., Dausch, G., and Ramakrishnan, I. V. More than meets the eye: A survey of screen-reader browsing strategies. In *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility* (2010), W4A '10, ACM, pp. 13:1–13:10.

[4] De Pascale, M., Mulatto, S., and Prattichizzo, D. Bringing haptics to Second Life. In *Proceedings of the 2008 Ambi-Sys Workshop on Haptic User Interfaces in Ambient Media Systems* (2008), HAS '08, ICST, pp. 6:1–6:6.

[5] Duddington, J. eSpeak: Speech synthesizer. `http://espeak.sourceforge.net`.

[6] Ferati, M., Mannheimer, S., and Bolchini, D. Acoustic interaction design through "audemes": experiences with the blind. In *Proceedings of the 27th ACM International Conference on Design of Communication* (2009), SIGDOC '09, ACM, pp. 23–28.

[7] Folmer, E., Yuan, B., Carr, D., and Sapre, M. TextSL: a command-based virtual world interface for the visually impaired. In *Proceedings of the 11th*

International ACM SIGACCESS Conference on Computers and Accessibility (2009), ASSETS '09, ACM, pp. 59–66.

[8] GMAGames. Shades of Doom. `http://gmagames.com/sod.shtml`, 2008.

[9] Gomila, L. Simple and fast multimedia library. `http://www.sfml-dev.org`.

[10] Iglesias, R., Casado, S., Gutierrez, T., Barbero, J. I., Avizzano, C., Marcheschi, S., and Bergamasco, M. Computer graphics access for blind people through a haptic and audio virtual environment. In *Proceedings of the 3rd IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications* (2004), IEEE, pp. 13–18.

[11] Kruger, R., and Van Zijl, L. Rendering virtual worlds in audio and text. In *Proceedings of the International Workshop on Massively Multiuser Virtual Environments* (2014), MMVE '14, ACM, pp. 5:1–5:2.

[12] Linden Labs. Second Life Official Site. `http://www.secondlife.com`.

[13] Loy, G. *Musimathics: The Mathematical Foundations of Music*. The MIT Press, Cambridge, Massachusetts, 2007.

[14] Microsoft Corporation. Microsoft Active Accessibility. `http://msdn.microsoft.com/en-us/library/ms971350.aspx`.

[15] Microsoft Corporation. Microsoft Speech API (SAPI) 5.4. `http://msdn.microsoft.com/en-us/library/ee125663(v=vs.85).aspx`.

[16] Miranda, E. *Computer Sound Design: Synthesis techniques and programming*, 2 ed. Focal Press, Linacre House, Jordan Hill, Oxford, 2002.

[17] Open Metaverse Foundation. libOpenMetaverse. `http://www.openmetaverse.org`.

[18] OpenSim Contributors. OpenSim. `http://www.opensimulator.org`.

[19] Radegast Development Team. Radegast. `http://radegast.org/`.

[20] Röber, N., and Masuch, M. Interacting with sound: An interaction paradigm for virtual auditory worlds. In *Proceedings of the International Conference on*

Auditory Display (2004), ICAD '04, Georgia Institute of Technology: International Community for Auditory Display.

[21] Röber, N., and Masuch, M. Leaving the screen: New perspectives in audio-only gaming. In *Proceedings of the International Conference on Auditory Display* (2005), ICAD '05, pp. 92–98.

[22] Rumsey, F. *Spatial Audio*. Focal Press, 2001.

[23] The Linux Foundation. atk/at-spi. `http://www.linuxfoundation.org/collaborate/workgroups/accessibility/atk/at-spi`.

[24] Trewin, S., Hanson, V., Laff, M. R., and Cavender, A. PowerUp: An accessible virtual world. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility* (2008), ASSETS '08, ACM, pp. 177–184.

[25] W3C. Accessible Rich Internet Applications (WAI-ARIA) 1.0. `http://www.w3.org/TR/wai-aria/`.

[26] W3C. Web Content Accessibility Guidelines (WCAG) 2.0. `http://www.w3.org/TR/WCAG20/`.

[27] Westin, T. Game accessibility case study: Terraformers—a real-time 3D graphic game. In *Proceedings of the 5th International Conference on Disability, Virtual Reality and Associated Technologies* (2004), pp. 95–100.

[28] White, G., Fitzpatrick, G., and McAllister, G. Toward accessible 3D virtual environments for the blind and visually impaired. In *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts* (2008), DIMEA '08, ACM, pp. 134–141.